

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of :
Satoshi INAMI et al. :
Serial No. NEW : **Attn: APPLICATION BRANCH**
Filed July 30, 2003 : Attorney Docket No. 2003-1067A

INFORMATION PROCESSING TERMINAL
AND INFORMATION PROCESSING
METHOD

CLAIM OF PRIORITY UNDER 35 USC 119

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

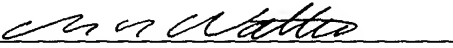
Sir:

Applicants in the above-entitled application hereby claim the date of priority under the International Convention of Japanese Patent Application No. 2002-222787, filed July 31, 2002, as acknowledged in the Declaration of this application.

A certified copy of said Japanese Patent Application is submitted herewith.

Respectfully submitted,

Satoshi INAMI et al.

By 
Charles R. Watts
Registration No. 33,142
Attorney for Applicants

CRW/krI
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
July 30, 2003

THE COMMISSIONER IS AUTHORIZED
TO CHARGE ANY DEFICIENCY IN THE
FEES FOR THIS PAPER TO DEPOSIT
ACCOUNT NO. 23-0975

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日
Date of Application:

2002年 7月31日

出 願 番 号
Application Number:

特願2002-222787

[ST.10/C]:

[JP2002-222787]

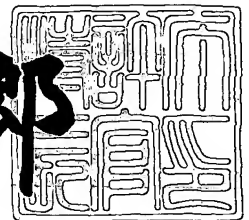
出 願 人
Applicant(s):

松下電器産業株式会社

2003年 6月 2日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2003-3042662

【書類名】 特許願

【整理番号】 2037340017

【提出日】 平成14年 7月31日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 13/10
G06F 13/18

【発明者】

【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

【氏名】 稲見 聡

【発明者】

【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

【氏名】 ダニエル ウェーバー

【発明者】

【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

【氏名】 福嶋 秀晃

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100097445

【弁理士】

【氏名又は名称】 岩橋 文雄

【選任した代理人】

【識別番号】 100103355

【弁理士】

【氏名又は名称】 坂口 智康

【選任した代理人】

【識別番号】 100109667

【弁理士】

【氏名又は名称】 内藤 浩樹

【手数料の表示】

【予納台帳番号】 011305

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9809938

【書類名】 明細書

【発明の名称】 競合解決処理装置

【特許請求の範囲】

【請求項 1】 デバイスが使用されているかどうか使用状態を管理する使用状態判定部と、アプリケーションやミドルウェアからなるリソースの使用要求を行う実行部と、使用しているアプリケーションの優先度を管理する使用中ソフトウェア優先度管理部と、リソースの使用要求を行っているアプリケーションの優先度を取得する要求ソフトウェア優先度取得部と、前記使用中ソフトウェア優先度管理部から得た優先度と、前記要求ソフトウェア優先度取得部とから得た優先度とを比較し、どのアプリケーションが優先されるかを判定する競合判定部と、リソースが空いており、前記使用状態判定部から得られる結果と前記競合判定部の結果から、使用要求を行っているアプリケーションの優先度が現在使用しているアプリケーションよりも高い場合、リソースへのアクセスを許可するとともに、アクセス要求を通知し、それ以外の場合は前記実行部へエラーを返すリソースアクセス部と、前記実行部からの使用要求を前記リソースアクセス部を通じてアクセス要求として受け付け、デバイスを制御するリソース部とを備える競合解決処理装置。

【請求項 2】 前記リソースアクセス部は、リソースを奪われたアプリケーションが再度リソースへのアクセスを要求した場合に、リソースが奪われたことを判断し、その旨をアプリケーションに通知する請求項 1 に記載の競合解決処理装置。

【請求項 3】 前記リソースアクセス部は、リソースを優先度の高いアプリケーションが優先度の低いアプリケーションから奪う場合に、現在実行中の処理をキャンセルした後で、リソースごとにリソースのリセットが必要かどうかを判断し、リセットが必要な場合にはリセット処理をしたあとで、リソースにアクセスする請求項 1 に記載の競合解決処理装置。

【請求項 4】 前記競合判定部は、アプリケーションがリソースに対する暗号化された優先度を持っていた場合に、認証部によって暗号化された優先度を復号化し、得られた優先度を基に判定を行う請求項 1 に記載の競合解決処理装置。

【請求項 5】前記競合判定部は、リソースに対してある優先度を持つアプリケーションが前記優先度とは異なる優先度を持つモジュールを 1 つ以上ロードあるいはリンクして動作する場合に、アプリケーションとモジュールの優先度の内、現在実行中のコードが属するモジュールの優先度を求め、それぞれの優先度に基づいて優先度の判定を行う請求項 1 に記載の競合解決処理装置。

【請求項 6】前記競合判定部は、すべてのアプリケーション制御を行う制御部から、リソースにアクセスするための優先度を得て判定を行う請求項 1 に記載の競合解決処理装置。

【発明の詳細な説明】

【 0 0 0 1 】

【発明の属する技術分野】

本発明は、リソースの競合解決処理装置に関し、より特定的には、複数のプログラムの内、優先度の高い特定のプログラムにスピーカなどのリソース獲得権を優先的に与える情報処理装置に関する。

【 0 0 0 2 】

【従来技術】

従来、携帯電話などの情報処理機器において、特定のアプリケーション、例えば電話の機能を他のアプリケーション、たとえばウェブブラウザなどより優先し、スピーカなどのリソースを電話アプリケーションに優先的に割り当て、たとえばウェブブラウザによりコンテンツを表示中であったとしても、電話の着信音を鳴動させることが行われている。これは携帯電話においては、電話の着信があった場合には、いつでもその処理を優先させ、通話を実現しなければならないためである。

【 0 0 0 3 】

優先的にリソースを割り当てる手法の第 1 の方法では、それぞれのリソースに対してどのアプリケーションをどのような優先度で優先するのか知っていて、すべてのアプリケーションを制御するソフトウェアコンポーネントが存在する。そして、この中央集権的なコンポーネントがリソースを使用したいアプリケーションのリソースアクセス要求を受け付けて、リソースが空きであるか、あるいは空

きでなくても現在使用中のアプリケーションよりも優先度が高ければ、リソースを占有しているアプリケーションにそのリソースを開放するように要求し、リソースが空きであることを保証してリソースアクセスを許可する。あるいは、中央のコンポーネントにより優先度を制御するのでは、実行速度的に問題がある特別な場合が存在したときには、リソースにアクセスする特別な関数を用意し、その関数の中で、例えばどのようなアプリケーションが使用中でも電話アプリケーションが必ずそのリソースを優先的に取得するように指示して、アドホックな対応で優先度を管理していた。このようにして第1の方法ではリソースの競合解決を行っている。

【0004】

また、第2の方法が、特許第2828971号公報に開示されている。第2の方法では、印刷のジョブを制御するのにキューが用いられている。またユーザの入力により、そのキューにあるジョブの順番を入れ替えることができる。従って、ユーザが優先させたいジョブはキューの先頭につなぐことができる。

【0005】

【発明が解決しようとする課題】

しかしながら、第1の方法のみで競合を解決しようとした場合、以下の問題点がある。つまり、第1の方法では、システムで動作するすべてのアプリケーションがリソース競合を解決するソフトウェアコンポーネントの競合解決の仕組みを理解し、その仕組みに従って作られなければならない。その仕組みに従って作られていないアプリケーションが存在する場合には、優先度の高いアプリケーションがリソースを要求した場合に、リソースを占有したままになってしまう可能性があるからである。また、リソースを優先度の高いアプリケーションに奪われる優先度の低いアプリケーションは、いつでもリソースを開放せよとの命令を受けた場合に、リソースを開放しなければならず、アプリケーション内部の構成が複雑になる。更に、中央のコンポーネントで制御しきれない処理については、例えば電話アプリケーション専用の特別な関数を用意して、アドホックな対応で管理するほかなく、ソフトウェアを再利用することが難しかった。

【0006】

また、第2の方法で競合を解決しようとした場合、アプリケーションに対してユーザがキューに対する処理を行わなければならない。すなわちアプリケーションはキューを保持して印刷のジョブを行っているプロセスに対してキャンセルを行い、次のジョブを行うように指示しなければならない。従って、優先度のことを考慮し、自動的に優先度の高いアプリケーションの処理を優先的に処理するということができない。また、要求をキューで管理しているので、リソース管理部の処理は複雑になるという問題が生じる。

【0007】

本発明は、上記問題点を解決するためになされたものであり、ある特定のアプリケーションが優先的にリソースを獲得できなければならない情報処理機器において、リソース競合解決の仕組みを知ることなく作られたアプリケーションが存在する場合においても、優先度の高いアプリケーションがリソースを要求した時にリソース使用中の優先度の低いアプリケーションがリソースを開放し、システムが期待通りに動作することが可能であり、かつ、リソースを優先度の高いアプリケーションに奪われる時には、リソースを奪われる側のアプリケーションがリソースの開放を意識することなくリソースのアクセス権の委譲が行われることを目的としている。更にリソース部へアクセスする関数およびリソース部のプログラムを汎用的に記述することで、それらが再利用できるようにすることである。

【0008】

【課題を解決するための手段】

上記目的を達成するために、本発明の競合解決処理装置は、デバイスが使用されているかどうか使用状態を管理する使用状態判定部と、アプリケーションやミドルウェアからなるリソースの使用要求を行う実行部と、使用しているアプリケーションの優先度を管理する使用中ソフトウェア優先度管理部と、リソースの使用要求を行っているアプリケーションの優先度を取得する要求ソフトウェア優先度取得部と、前記使用中ソフトウェア優先度管理部から得た優先度と、前記要求ソフトウェア優先度取得部とから得た優先度とを比較し、どのアプリケーションが優先されるかを判定する競合判定部と、リソースが空いており、前記使用状態判定部から得られる結果と前記競合判定部の結果から、使用要求を行っているアプ

리케이션の優先度が現在使用しているアプリケーションよりも高い場合、リソースへのアクセスを許可するとともに、アクセス要求を通知し、それ以外の場合は前記実行部へエラーを返すリソースアクセス部と、前記実行部からの使用要求を前記リソースアクセス部を通じてアクセス要求として受け付け、デバイスを制御するリソース部とを備えることを特長とする。

【 0 0 0 9 】

【発明の実施の形態】

以下、本発明の実施の形態について、図面を用いて詳細に説明する。

【 0 0 1 0 】

(実施の形態 1)

図 1 は、本発明の実施の形態における競合解決処理装置の構成の一例を示すブロック図である。実施の形態 1 において、携帯端末などの情報処理装置が競合解決処理装置を備えているものとする。この競合解決処理装置は、リソースアクセス部 1 1 と、使用状態判定部 1 2 と、競合判定部 1 3 と、使用中ソフトウェア優先度管理部 1 4 と、要求ソフトウェア優先度取得部 1 5 と、リソース部 1 6 と、実行部 1 7 とから構成される。

【 0 0 1 1 】

リソース部 1 6 は、携帯端末におけるデバイスであって、例えばスピーカや液晶ディスプレイなどの表示画面などである。実行部 1 7 はプログラムであって、より具体的には、アプリケーション、あるいはアプリケーションにくみ込み可能なライブラリやその他のミドルウェア、あるいはドライバである。

【 0 0 1 2 】

図 2 は、図 1 の構成を有する競合解決装置の動作を示すフローチャートである。なお、図 2 において、実行部 1 7 はアプリケーション A とアプリケーション B とし、アプリケーション A よりも優先度が高いアプリケーション B がリソースの使用要求（アクセス要求）を行うものとする。

【 0 0 1 3 】

また、リソース部 1 6 は、リソース部 1 6 をアクセスするための関数群をあらかじめリソースアクセス部 1 1 に登録してあるとする。この登録はリソースアク

セス部 1 1 が提供する汎用的なインタフェースにより行われるため、リソース部 1 6 作成者はそのガイドラインにさえ従っていれば良く、第 3 者がこのリソース部 1 6 を作成することが容易となる。

【 0 0 1 4 】

まず、ユーザの操作を契機として、アプリケーション A よりも優先度が高いアプリケーション B がリソース部 1 6 を使用するために、リソースアクセス部 1 1 に対してアクセス要求を行い、処理を開始する（S 1）。

【 0 0 1 5 】

次に、リソースアクセス部 1 1 は、アプリケーション B からのアクセス要求を受信し、アクセスしようとしている対象のリソースを認識し、そのリソースに対して現在他のアプリケーションが使用中かどうか、使用状態判定部 1 2 に対して問い合わせを行う。使用状態判定部 1 2 が指定されたリソースに対してアクセスをしているアプリケーションがあるかどうかを判断し、リソースアクセス部 1 1 に解析結果を返す（S 2）。リソースアクセス部 1 1 は、解析結果を受けて、現在使用中のアプリケーションがなければ（S 2 が未使用）、アプリケーション B に対してリソースのアクセスを許可し、リソース部 1 6 にアプリケーション B の要求を行う（S 7）。このとき、あらかじめリソース部 1 6 が登録しておいた関数が使用される。

【 0 0 1 6 】

一方、解析結果により現在使用中のアプリケーションがあれば（S 2 が使用）、競合を解決するために、リソースアクセス部 1 1 は、競合判定部 1 3 にアプリケーション B の情報を渡し、判定を要求する。競合判定部 1 3 は、まず現在使用中のアプリケーションの優先度を使用中ソフトウェア優先度管理部 1 4 に問い合わせ、アプリケーション A のリソースに対する優先度を得る（S 3）。次に、競合判定部 1 3 は、要求ソフトウェア優先度取得部 1 5 に問い合わせ、アプリケーション B のリソースに対する優先度を得る（S 4）。

【 0 0 1 7 】

競合判定部 1 3 は、使用中ソフトウェア優先度管理部 1 4 から得られたアプリケーション A の優先度と、要求ソフトウェア優先度取得部 1 5 から得られたアプ

리케이션 B の優先度を比較し、その結果をリソースアクセス部 1 1 に返す。リソースアクセス部 1 1 は、競合判定部 1 3 から受け取った結果を基に、処理を決定する (S 5)。

【 0 0 1 8 】

リソースアクセス部 1 1 は、アプリケーション B の優先度が低ければリソースへのアクセスを拒否し (S 5 がアクセスを拒否)、アプリケーション B にアクセスエラーを返す (S 6)。一方、アプリケーション B の優先度が高ければリソースへのアクセスを許可し (S 5 がアクセスを許可)、リソース部 1 6 に対して、あらかじめ登録されている関数をコールし、アプリケーション A の処理をキャンセルするように要求した後で、アプリケーション B の要求を行う (S 7)。

【 0 0 1 9 】

以上により、アプリケーション A やアプリケーション B は、単に必要なリソースに対して実行を要求 (アクセス要求) するのみで、リソース管理を専門とするコンポーネントに対してリソース競合の調停を要求するなどの処理は必要としない。しかし、あらかじめアプリケーション A とアプリケーション B の優先度について、アプリケーション A の優先度の方が B に比べて低くなるように設定しておけば、例えばアプリケーション B が緊急に音を鳴動したい場合などにアプリケーション A が使用中であったとしても、簡単にスピーカのリソースをアプリケーション B に渡すことができる。また、状況が変わり、アプリケーションの優先度が変わった場合でも、その設定を変更するのみでよく、アプリケーションや、リソース部 1 6 を変更する必要はない。またリソース部 1 6 の実行関数としては、単に処理のキャンセルを行う関数や、実行の関数を作成し、リソースアクセス部 1 1 に登録しておくだけでよい。

【 0 0 2 0 】

従って、競合解決処理装置によれば、アプリケーション作成者がリソース競合解決について意識しつつアプリケーションを作成しなくても、アプリケーションの優先度をアプリケーションと関連付けて管理することにより、リアルタイムシステムなどで必要となるリソースの瞬時の委譲を行うことが可能となる。また、リソース部 1 6 は、汎用的な枠組みでリソース部 1 6 を作成可能であり、複数の

リソース部 16 の再利用性が高くなる。

【0021】

なお、図 2 では、実行部 17 としてアプリケーションを仮定したが、アプリケーションでなくても、単にアプリケーションから要求を受けたミドルウェアや、ドライバなどのソフトウェアであってもよい。これについては、後述する。

【0022】

また、図 2 では、S3 と S4 において、使用中ソフトウェア優先度管理部 14 と要求ソフトウェア優先度取得部 15 は、単にアプリケーションの情報から優先度を判定すると記述したが、使用中ソフトウェア優先度管理部 14 と要求ソフトウェア優先度取得部 15 は、アプリケーションの種類とリソースの種類による 2 次元のマトリックスを持っていて、そのテーブルから優先度を判定してもよい。

【0023】

この場合、優先度を判定するためのアプリケーションの種類としては、アプリケーションの名前、アプリケーションごとに振られるアプリケーション ID、アプリケーションをグループ化した場合にアプリケーショングループごとに振られるアプリケーショングループ ID、アプリケーションを実行しているプロセスやスレッドの ID などの情報を使用してもよい。このマトリックスの例を図 3 に示す。

【0024】

このようにテーブルで管理することで、簡単に優先度の設定を変更ができ、将来優先度を変更した場合にもアプリケーションを変更しなくてもよい。

【0025】

あるいは優先度の判定は、アプリケーションが使用しているリソースの数や時間を計測しておき、それらの情報をもとに行ってもよい。そうすることで、ある少数のアプリケーションがリソースを使用しつづけたり、多数のリソースを占有したりするといったことを防ぐことができる。

【0026】

また、使用中ソフトウェア優先度管理部 14 は、単に競合判定部 13 が判定をした後や、リソースアクセス部 11 が実際にリソース部 16 にアクセスした後で

、使用中ソフトウェア優先度管理部 1 4 に現在使用中のアプリケーションの優先度を通知してもらい、優先度のみを管理してもよい。

【 0 0 2 7 】

なお、リソースとしては、スピーカの例をあげたが、LED、ディスプレイなどであってもよいし、具体的なデバイスでなくとも通信コネクションのような抽象的なリソースであってもよい。

【 0 0 2 8 】

なお、優先度の判定は、高いか、低いかのみを記述したが、優先度が同じ場合も考えられ、その場合は、高い場合の処理か低い場合の処理のどちらかを前もって決めておき、そのルールに従えばよい。

【 0 0 2 9 】

図 4 は、図 1 の構成を有する競合解決装置の動作を示すフローチャートである。なお、図 4 において、以前アプリケーション A はリソースを使用していて、その後アプリケーション B の要求によりリソースが奪われたものとする。この場合、使用状態判定部 1 2 にアプリケーション A が使用中に奪われたという情報を記録しておく。

【 0 0 3 0 】

まず、アプリケーション A は、再度リソースを使用するために、リソースアクセス部 1 1 に対してアクセスを要求する (S 1 1) 。

【 0 0 3 1 】

次に、リソースアクセス部 1 1 は、アプリケーション A からのアクセス要求を受信し、アクセスしようとしている対象のリソースを認識し、そのリソースに対して現在他のアプリケーションが使用中かどうか、また、使用中に奪われたかどうかを使用状態判定部 1 2 に対して問い合わせを行う。使用状態判定部 1 2 が指定されたリソースに対してアクセスをしているアプリケーションがあるかどうかを判断し、判断結果と、アプリケーション A が使用中に奪われたかどうかをリソースアクセス部 1 1 に返す (S 1 2) 。

【 0 0 3 2 】

リソースアクセス部 1 1 は、使用状況判定部 1 2 からの応答を受けて、現在要

求したリソースが使用されていなければ（S 1 2 が未使用）、使用状況判定部 1 2 に通知してアプリケーション A が使用中に奪われたという情報をクリアし、アクセスを許可し、かつアプリケーション A に以前リソースが使用中に奪われたことを通知する（S 1 7）。

【0 0 3 3】

一方、解析結果により現在使用中のアプリケーションがあれば（S 1 2 が使用）、競合を解決するために、リソースアクセス部 1 1 は、競合判定部 1 3 にアプリケーション A の情報を渡し、判定を要求する。競合判定部 1 3 は、まず現在使用中のアプリケーションの優先度を使用中ソフトウェア優先度管理部 1 4 に問い合わせ、使用中アプリケーションのリソースに対する優先度を得る（S 1 3）。次に、競合判定部 1 3 は、要求ソフトウェア優先度取得部 1 5 に問い合わせ、アプリケーション A のリソースに対する優先度を得る（S 1 4）。

【0 0 3 4】

競合判定部 1 3 は、使用中ソフトウェア優先度管理部 1 4 から得られた愛用中アプリケーションの優先度と、要求ソフトウェア優先度取得部 1 5 から得られたアプリケーション A の優先度を比較し、その結果をリソースアクセス部 1 1 に返す。リソースアクセス部 1 1 は、競合判定部 1 3 から受け取った結果を基に、処理を決定する（S 1 5）。

【0 0 3 5】

リソースアクセス部 1 1 は、アプリケーション A の優先度が低ければリソースへのアクセスを拒否し（S 1 5 がアクセスを拒否）、アプリケーション A にアクセスエラーを返す（S 1 6）。一方、アプリケーション A の優先度が高ければリソースへのアクセスを許可し（S 1 5 がアクセスを許可）、使用状況判定部 1 2 に通知してアプリケーション A が使用中に奪われたという情報をクリアし、アクセスを許可し、かつアプリケーション A に以前リソースが使用中に奪われたことを通知する（S 1 7）。

【0 0 3 6】

以上により、リソースに再度アクセスした時に、以前リソース使用中に奪われていた場合は、その旨を通知することができる。この通知を受けることで、以前

リソースを使用していたときとリソースの状態が異なることを検出し、必要なリソースの初期化処理などを再度行うことが可能となる。

【 0 0 3 7 】

さて、図 4 では、アプリケーションが必要な初期化を行ったが、リソースごとに初期化が必要かどうかを判定できる場合がある。その場合の競合解決装置の動作を、図 5 に示すフローチャートを用いて説明する。

【 0 0 3 8 】

なお、図 5 の S 2 1 から S 2 4 までは、図 2 の S 1 から S 4 までと同じあり、S 2 4 では、同様に優先度の判定を行うべく優先度の取得を行う。S 2 5 以降の動作について、以下に説明する。

【 0 0 3 9 】

競合判定部 1 3 は、使用中ソフトウェア優先度管理部 1 4 から得られたアプリケーション A の優先度と、要求ソフトウェア優先度取得部 1 5 から得られたアプリケーション B の優先度を比較し、その結果をリソースアクセス部 1 1 に返す。リソースアクセス部 1 1 は、競合判定部 1 3 から受け取った結果と、リソースにアクセスする前にリソースがリセットを必要かどうか判定し、処理を決定する（S 2 5）。

【 0 0 4 0 】

リソースアクセス部 1 1 は、アプリケーション B の優先度が低ければリソースへのアクセスを拒否し（S 2 5 がアクセスを拒否）、アプリケーション B にアクセスエラーを返す（S 2 6）。一方、アプリケーション B の優先度が高く、リソースのリセットが不要である場合（S 2 5 が初期化不要）、リソース部 1 6 に対して、あらかじめ登録されている関数をコールし、アプリケーション A の処理をキャンセルするように要求した後で、アプリケーション B の要求を行う（S 2 8）。また、アプリケーション B の優先度が高く、リソースのリセットが必要である場合（S 2 5 が初期化要）、リソースの初期化を行い（S 2 7）、リソース部 1 6 にアクセスを行う（S 2 8）。

【 0 0 4 1 】

以上により、リソースごとにリセットが必要か判断することで、アプリケーション

ョンが初期化をするかどうか意識する必要がなくなる。

【0042】

なお、ここではリソースごとに判断を行ったが、リソースとアプリケーションの組み合わせにより、リセットを行うかどうかを判断しても良いし、ごく単純なシステムで、あらかじめリセットをしてよいことがわかっている場合は、すべての場合においてリセットを行ってもよい。

【0043】

また、リソースごとにリセットが必要かどうかは、リセットを行う関数のポインタが登録されているかどうかによって判定してもよい。

【0044】

次に、図2の説明で前述した、アプリケーションに組み込まれているライブラリや、モジュール毎に優先度を保有するケースについて説明する。

【0045】

アプリケーション毎にアプリケーションIDなどから優先度を取得すると前述したが、この場合、ライブラリや、モジュールごとに優先度を取得する。例えばライブラリ内で優先度を変えたい場合は、その処理ごとに優先度の異なるスレッドを作成したりしてもよい。あるいはライブラリの管理者IDなどの属性値より優先度を取得してもよい。

【0046】

また、リソースにアクセスするごとに使用ソフトウェア優先度管理部14に、優先度を記述することで、モジュールごとに優先度を変えながら処理をすることが可能となる。

【0047】

優先度としては、アプリケーションごとに含まれるモジュールの優先度の内、最高の優先度により比較してもよいし、最低の優先度により比較してもよいし、実行中のライブラリにあわせて優先度を変化させてもよい。

【0048】

以上のように、優先度を動的に変えることにより、例えば同じアプリケーションによっても優先度が高い状態で処理したい場合や、低い優先度でも良い場合な

どが混在した場合に、それぞれ最適な優先度で処理することができる。具体的には、電話の通話中には優先度を高くして、電話のメニューを表示している場合には優先度を低くするなど、リソースにアクセスするのに必要な優先度は異なる。

【 0 0 4 9 】

また、アプリケーションが信頼できないソフトウェアハウスによって作成されたライブラリを使用している場合、そのアプリケーション自体の信頼度が低下するために、そのライブラリの優先度にあわせて優先度を低くして動作することは、合理的であると考えられる。

【 0 0 5 0 】

(実施の形態 2)

図 6 は、本発明の実施の形態における競合解決処理装置の構成の一例を示すブロック図である。実施の形態 2 において、携帯端末などの情報処理装置が、競合解決処理装置を備えているものとする。この競合解決処理装置は、リソースアクセス部 1 1 と、使用状態判定部 1 2 と、競合判定部 1 3 と、使用中ソフトウェア優先度管理部 1 4 と、要求ソフトウェア優先度取得部 1 5 と、リソース部 1 6 と、実行部 1 7 と、暗号解析部 1 8 とから構成される。実施の形態 1 との差異は、アプリケーションの優先度が暗号化されてアプリケーションに組み込まれている点であり、その他は同じである。図 6 においては、暗号解析部 1 8 が、図 1 に追加されている。この暗号解析部 1 8 は、アプリケーションに組み込まれた暗号化された優先度を復号化し、優先度を取得する。

【 0 0 5 1 】

暗号解析部 1 8 を有する場合のフローチャートを、図 7 に示した。なお、S 1 から S 3 までは、図 2 の S 1 から S 3 までと同じあり、図 7 では省略した。また、S 4 から S 7 までは、図 2 の S 4 から S 7 までと同じあり、これも同様に図 7 では省略した。

【 0 0 5 2 】

S 3 で、リソースアクセス部 1 1 は、競合判定部 1 3 にアプリケーションの情報を渡し、判定を要求する。競合判定部 1 3 は、まず現在使用中のアプリケーションの優先度を使用中ソフトウェア優先度管理部 1 4 に問い合わせ、使用中アプ

리케이션のリソースに対する優先度を得る（S3）。ここで使用中ソフトウェア優先度管理部14から取得する優先度は、暗号化されているものとする。次に、競合判定部13は、暗号解析部18を用いて使用中アプリケーションの暗号化された優先度を複合化し（S33）、複合化に成功すると（S33が成功）、S4に遷移し、以下の処理は、図2と同様である。一方、複合化に失敗すると（S33が失敗）、S16に遷移し、アプリケーションが信頼できないとしてエラーを返す。

【0053】

以上のように、優先度の信頼性を高めることにより、不正なアプリケーションが、優先度を自らコントロールしようと勝手にアプリケーションIDを変更して使用した場合などに起こりえる不測の事態を避けることができる。

【0054】

（実施の形態3）

図8は、本発明の実施の形態における競合解決処理装置の構成の一例を示すブロック図である。実施の形態3において、携帯端末などの情報処理装置が、競合解決処理装置を備えているものとする。この競合解決処理装置は、リソースアクセス部11と、使用状態判定部12と、競合判定部13と、使用中ソフトウェア優先度管理部14と、要求ソフトウェア優先度取得部15と、リソース部16と、実行部17と、中央制御部21とから構成される。図8においては、中央制御部21が、図1に追加されている。この中央制御部21は、システム全体のリソースの整合をとるために導入されているものである。本実施の形態では、途中で優先度が低いアプリケーションにリソースを奪われたくないアプリケーションは必ず中央制御部21にリソースへのアクセスが可能か問い合わせを行うものとする。リソースが奪われてもいいアプリケーションは、中央制御部21を意識することなく動作し、実施の形態1で記述してきたとおりに、優先度の高いアプリケーションがリソースへのアクセスを要求した場合には、リソースを奪われることになる。

【0055】

中央制御部21がアプリケーションの状態を意識してリソース競合を解決する

場合の動作を、図9に示すフローチャートを用いて説明する。なお、図9において、実行部17はアプリケーションとする。

【0056】

まず、アプリケーションBは、ユーザの入力動作やネットワークからのイベントの通知によりリソースにアクセスすることが必要な場合、中央制御部21に対してアクセス可能かどうかの問い合わせを行う（S41）。

【0057】

問い合わせを受けた中央制御部21は、現在リソースをアクセス中のアプリケーションと、アプリケーションBの優先度を判断し（S42）、アプリケーションBがアクセス可能であれば（S42が可能）、その結果をアプリケーションB、すなわち実行部17に通知し（S43）、リソースアクセス部11に対してアプリケーションBがアクセス可能であることと、その優先度を通知する（S45）。以下、リソースアクセス部11は中央制御部21から通知された情報をもとに、競合解決を行う。これ以降のステップは、図2のS2以降と同じである。

【0058】

一方、中央制御部21からの通知がアクセス不可であった場合（S42が不可能）、アクセスを中止する（S44）。

【0059】

以上のように、中央制御部21に対してまずアプリケーションがリソースに対してアクセス可能かを問い合わせることにより、リソースアクセス部11など、これまで述べてきた競合解決システムの機能は、実施の形態1に比べて単純になる。すなわち、実施の形態1では、優先度のテーブルを管理していたが、実施の形態3では中央制御部21がすべてのテーブルを一括して制御可能となる。従って、管理するリソースが多数ある場合に、より効率のよい競合解決を行うことが可能となる。

【0060】

しかも、実施の形態1で述べた競合解決システムがなく、中央制御部21のみでリソース管理を行う場合に比べて、アプリケーションの構成は単純になる。すなわち、中央制御部21のみですべてのリソース管理を行う場合は、アプリケー

ションはリソースが奪われる時に、リソースの開放を要求される。また、リソースを解放した後で、中央制御部 2 1 に対して解放完了の通知を行わなければならない。また、リソースを要求し、リソース解放を待っているアプリケーションは、その解放処理が終わるまで処理を待つ状態が必要となり、アプリケーションの構成が複雑になる。さらに、すべてのアプリケーションは中央制御部 2 1 を意識して作成されなければならない。そうでないと、必要なときにリソースの解放が行われない場合があるからである。これに対して、中央制御部 2 1 がアクセス権の通知のみを行う本実施の形態では上記のような制限はない。

【 0 0 6 1 】

【発明の効果】

以上のように本発明によれば、競合解決部により優先度に基づきリソースのアクセス制御を行うことで、個々のアプリケーション作成時に、電話機能を優先することを考慮しなくてよくなり、アプリケーション作成者の負担が減少し、ソフトウェアの構成も容易となる。また、電話機能を優先するという特別なルールが必要なくなり、サードパーティがアプリケーションを作成する敷居が低くなり、より多くのアプリケーションが作成されることが期待される。また、あとでアプリケーション間の優先度が変化する場合にも、ルールを変更するだけでよく、アプリケーションの再利用性が向上する。更に、リソース部のプログラムはこの競合解決の枠組みに合う関数を作成し、リソースアクセス部に登録するだけでよく、リソース部の再利用性が高まる。

【図面の簡単な説明】

【図 1】

本発明の一実施の形態に係る競合解決処理装置の構成を示すブロック図

【図 2】

本発明の一実施の形態に係る競合解決処理装置の処理手順を示すフローチャート

【図 3】

本発明の一実施の形態に係る競合解決処理装置の優先度による競合解決に用いられるアプリケーション優先度管理テーブルの概要図

【図 4】

本発明の一実施の形態に係る競合解決処理装置の処理手順を示すフローチャート

【図 5】

本発明の一実施の形態に係る競合解決処理装置の処理手順を示すフローチャート

【図 6】

本発明の一実施の形態に係る競合解決処理装置の構成を示すブロック図

【図 7】

本発明の一実施の形態に係る競合解決処理装置の処理手順を示すフローチャート

【図 8】

本発明の一実施の形態に係る競合解決処理装置の構成を示すブロック図

【図 9】

本発明の一実施の形態に係る競合解決処理装置の処理手順を示すフローチャート

【符号の説明】

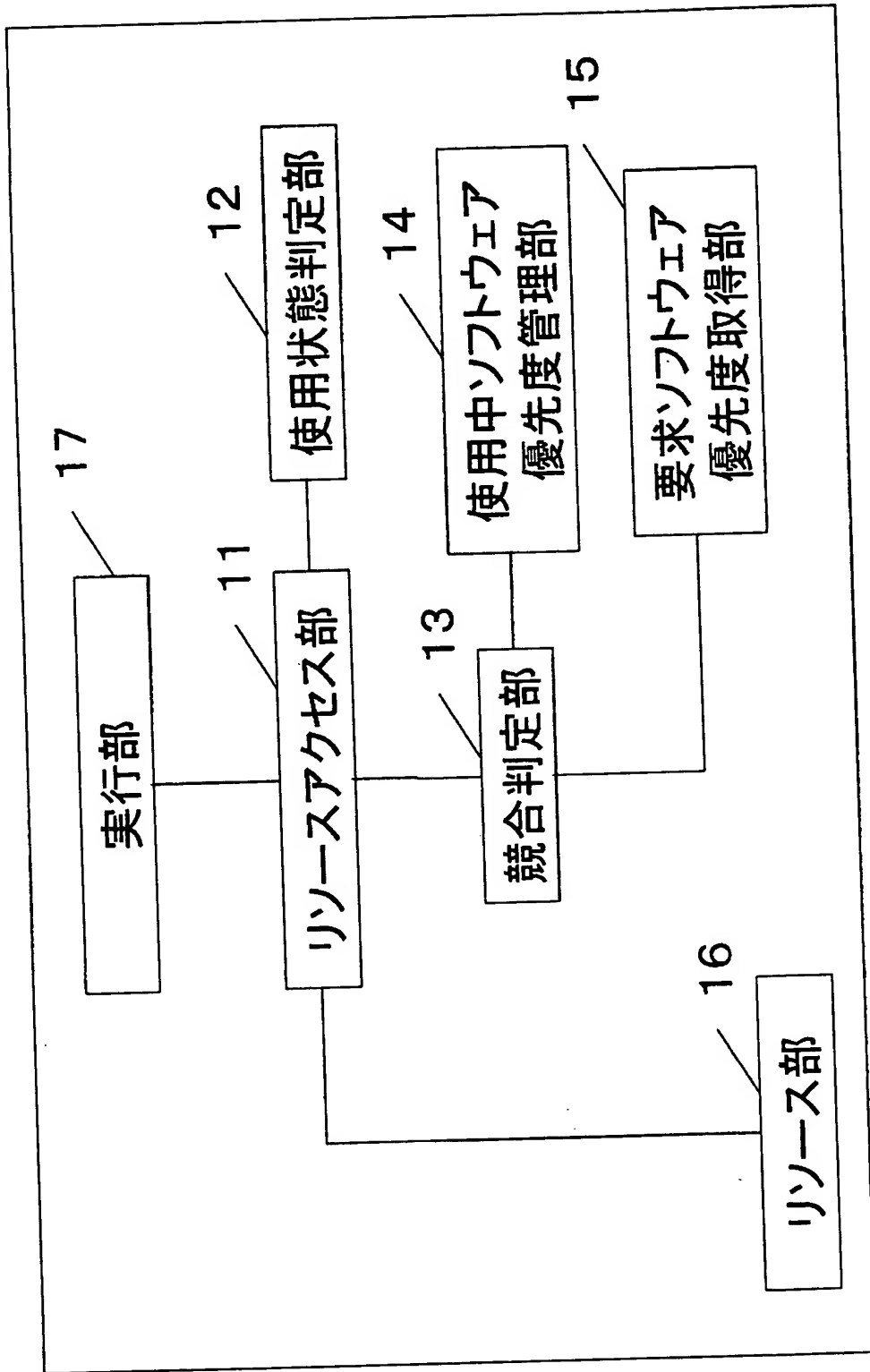
- 1 1 リソースアクセス部
- 1 2 使用状態判定部
- 1 3 競合判定部
- 1 4 使用中ソフトウェア優先度管理部
- 1 5 要求ソフトウェア優先度取得部
- 1 6 リソース部
- 1 7 実行部
- 1 8 暗号解析部
- 2 1 中央制御部

【書類名】

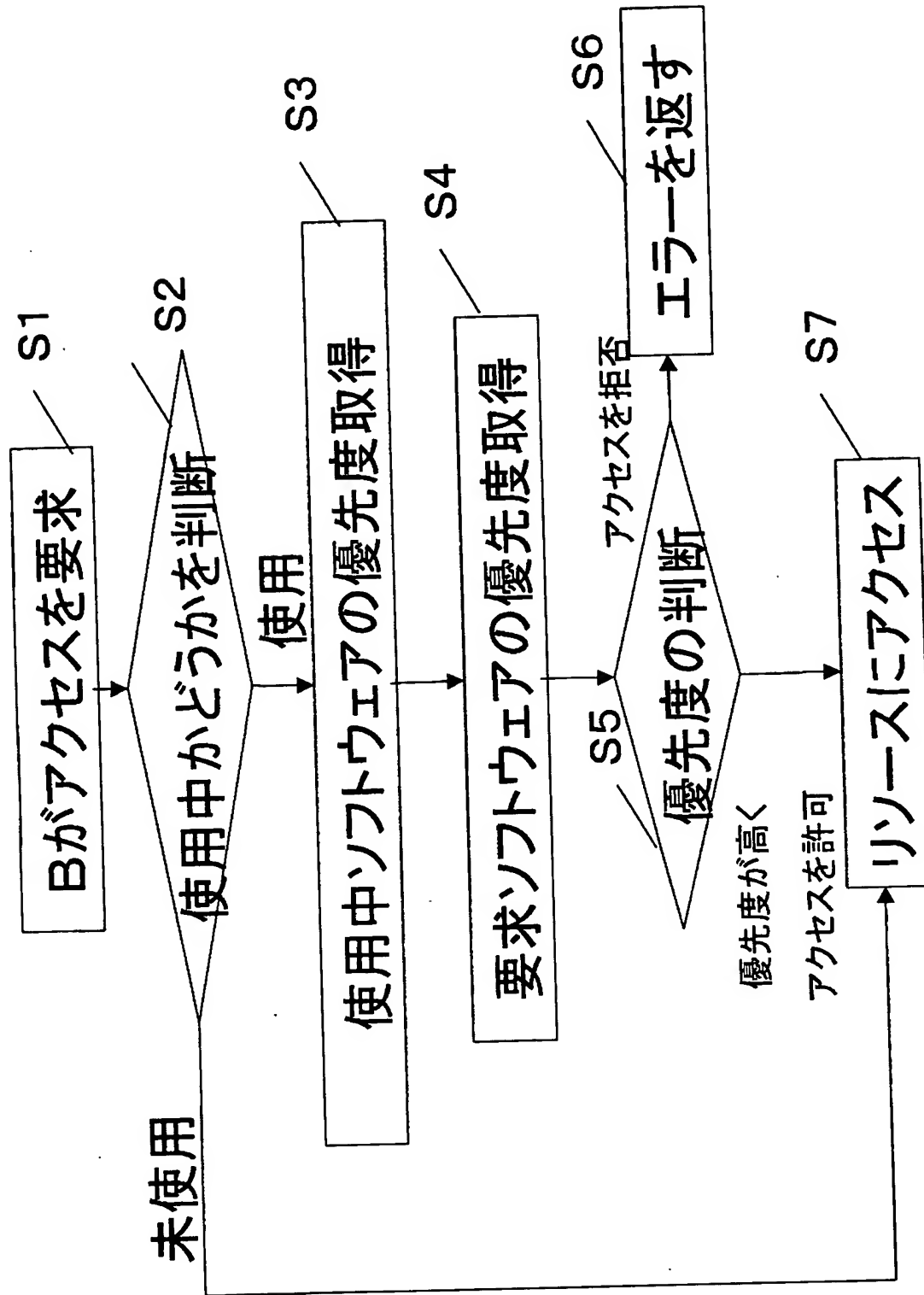
図面

【図 1】

携帯端末



【図 2】

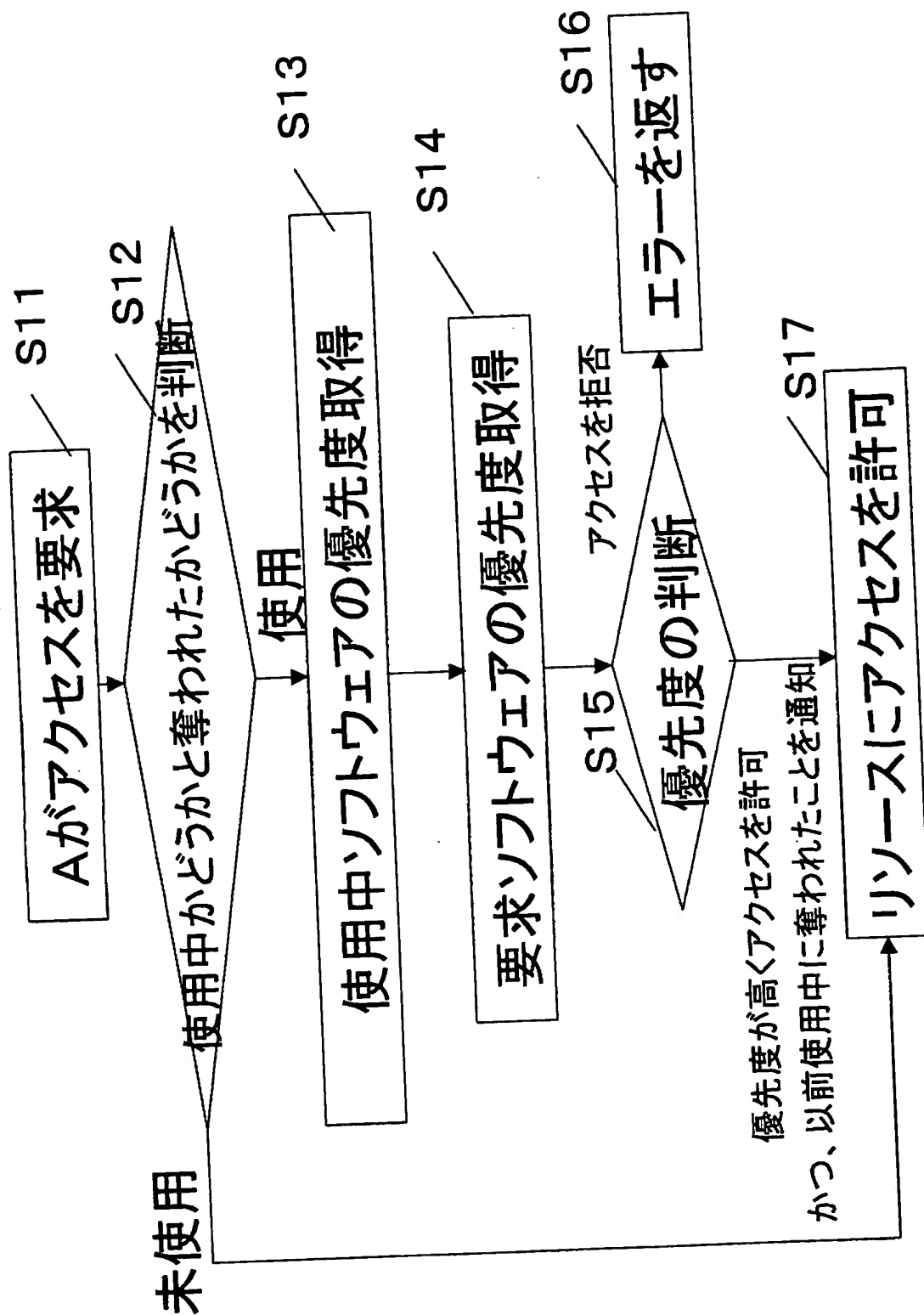


【図3】

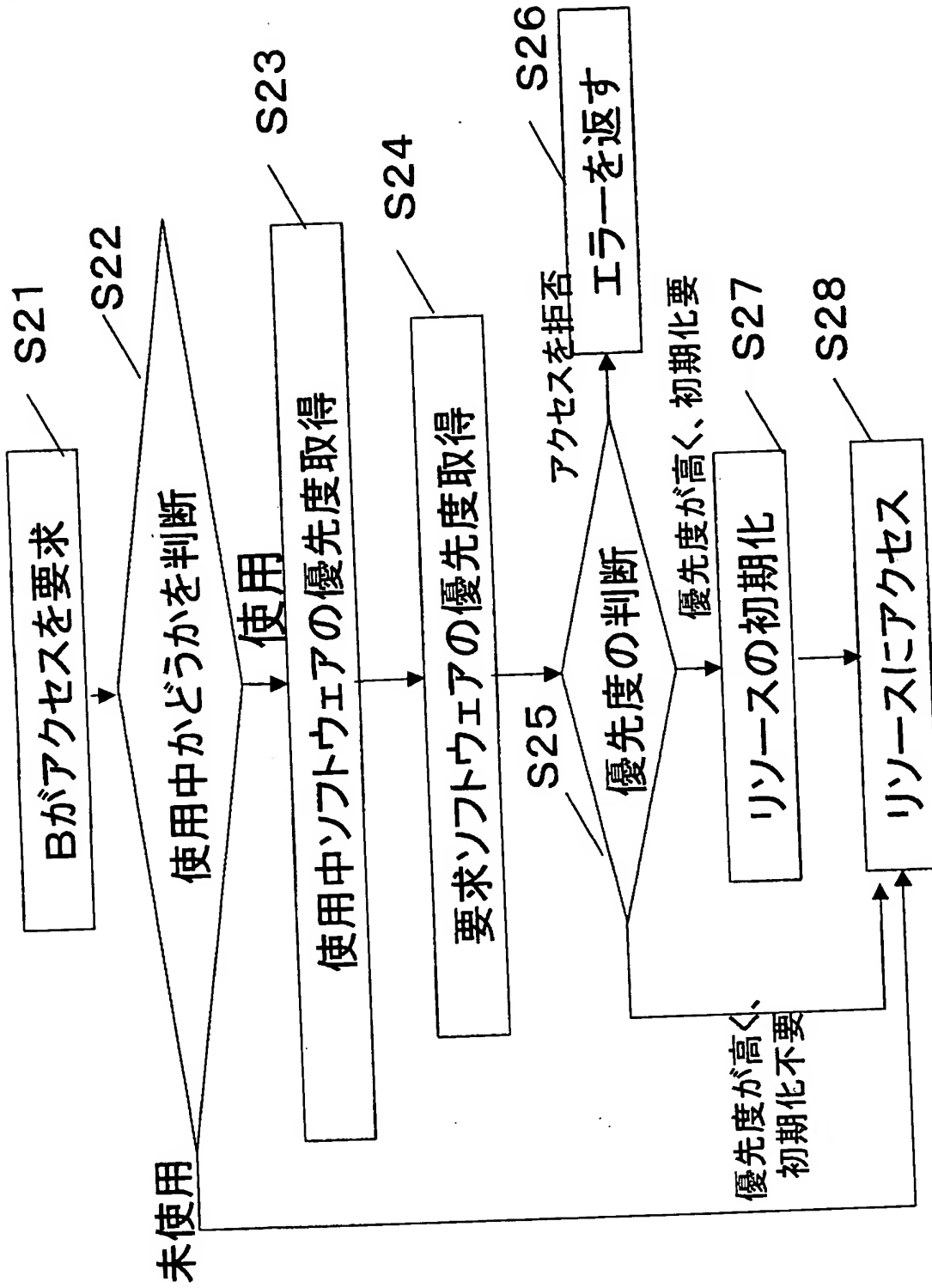
アプリケーション優先度管理テーブル

	ブラウザ	メール	電話
スピーカー	2	2	1
ディスプレイ	3	2	1
通信	1	1	2
コネクション			
LED	2	3	1

【図 4】

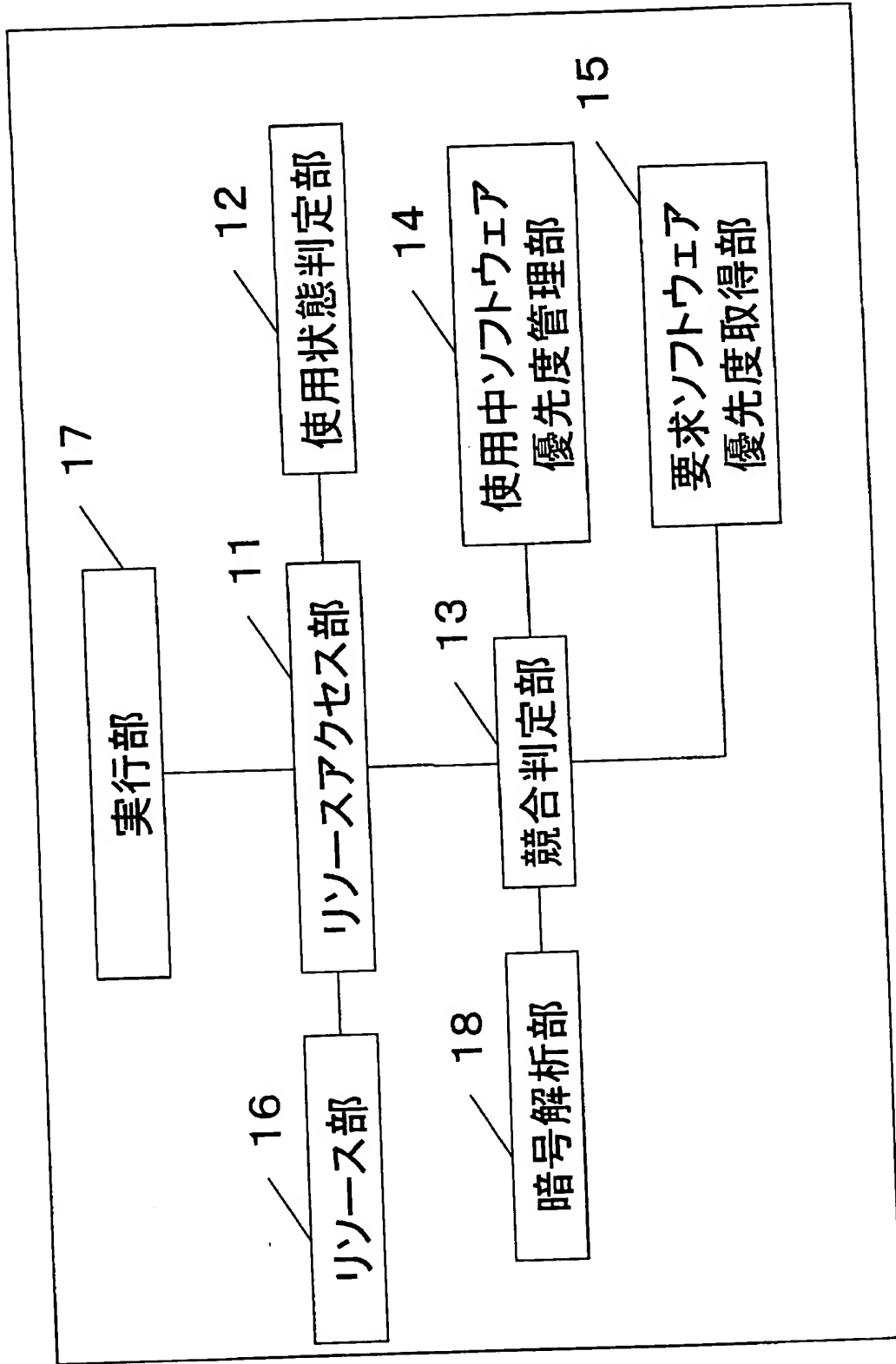


【図 5】

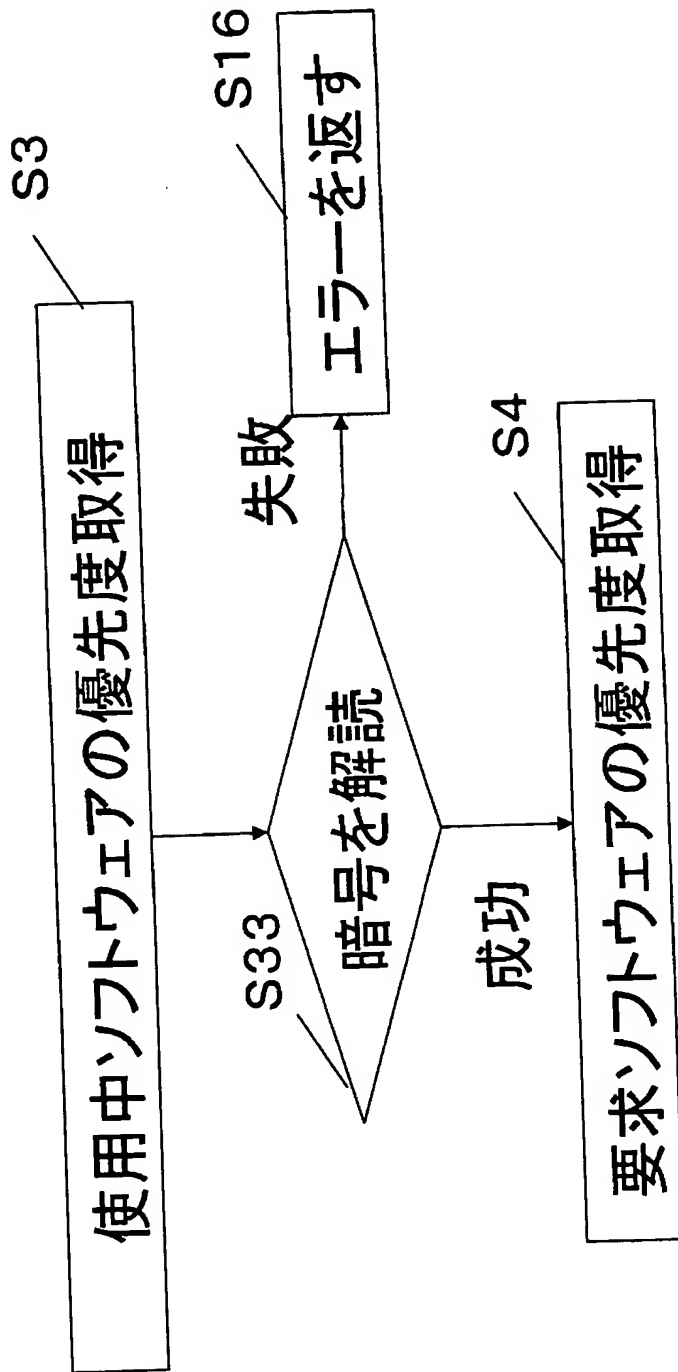


【図6】

携帯端末

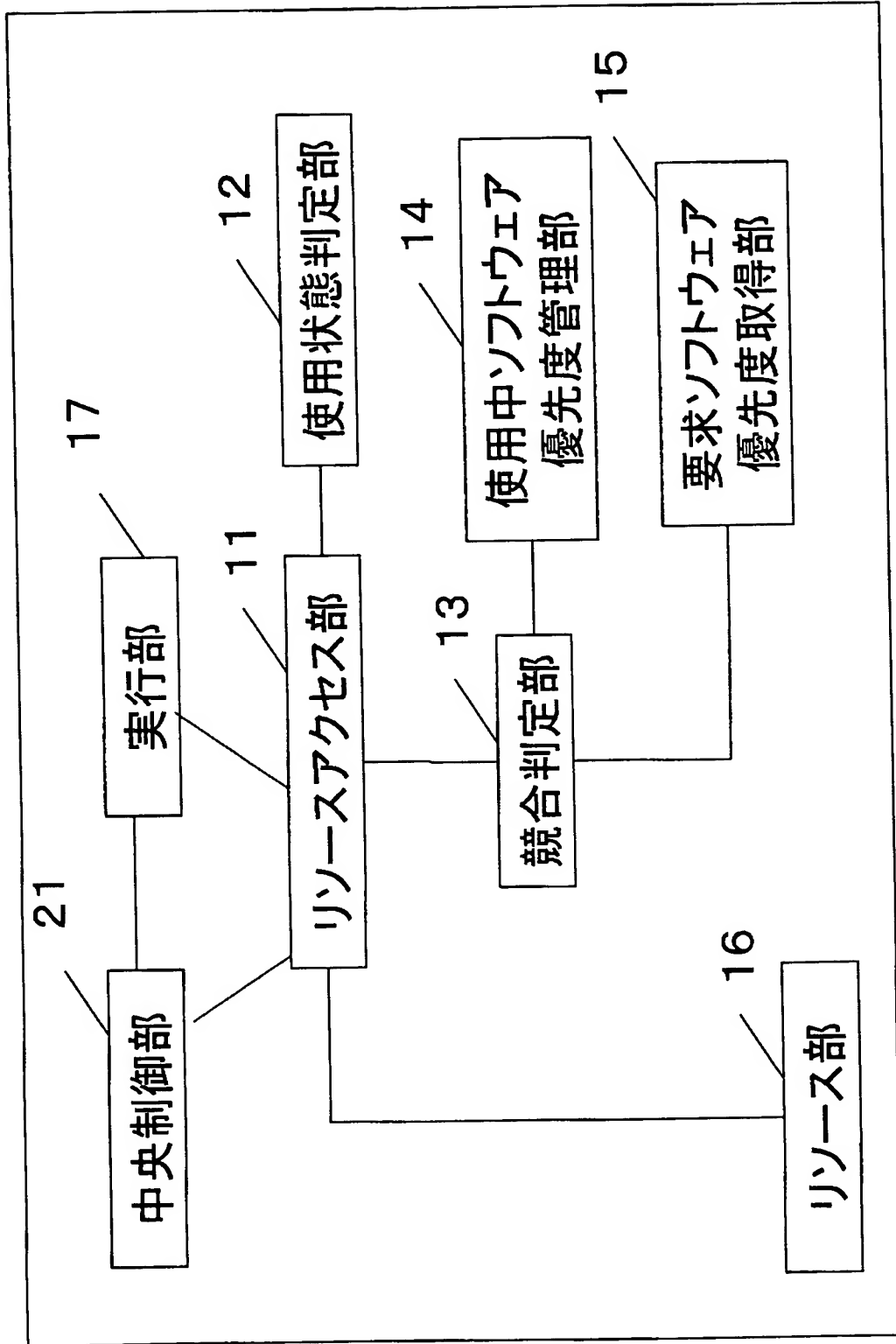


【図7】

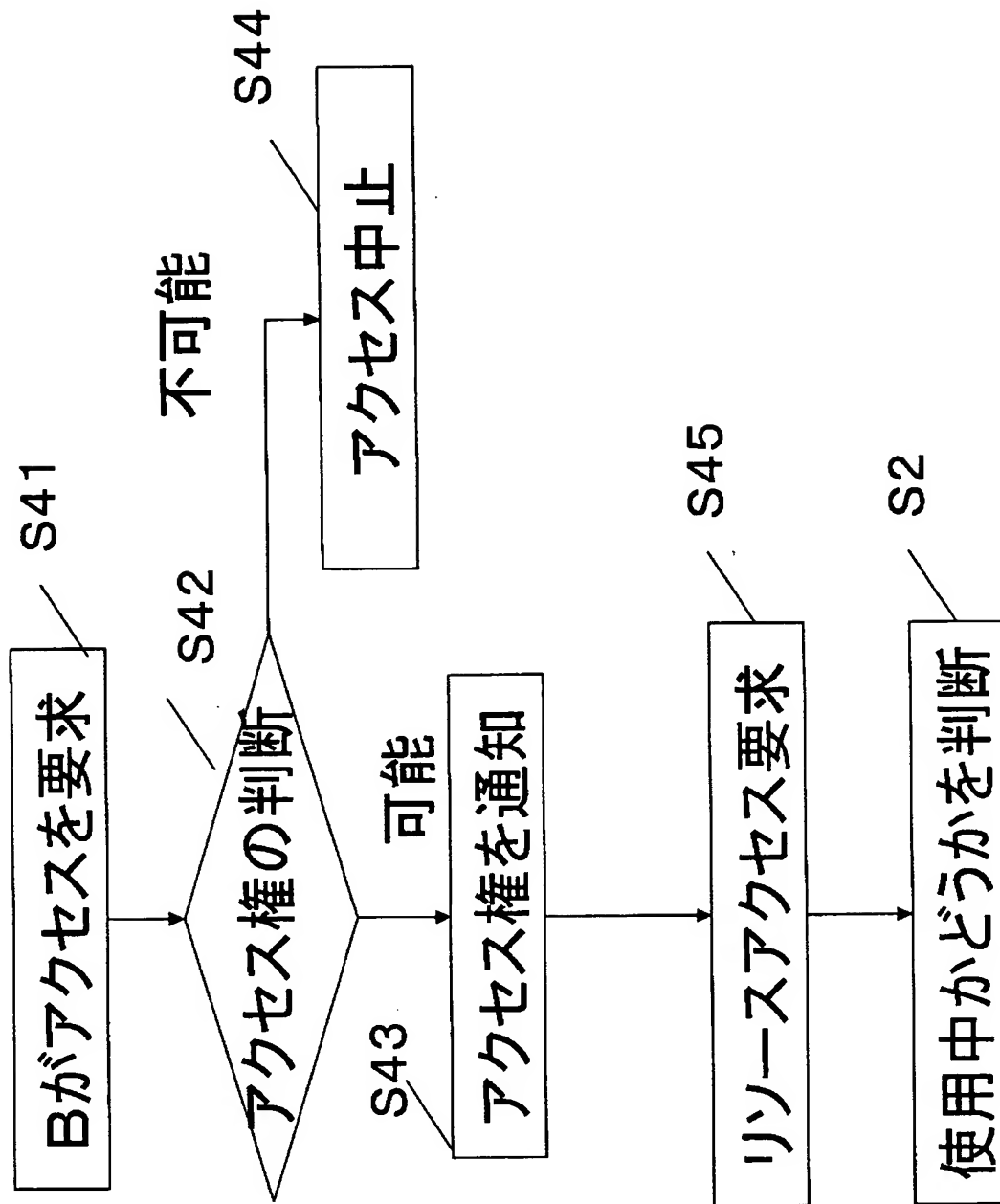


【図8】

携帯端末



【図 9】



【書類名】 要約書

【要約】

【課題】 アプリケーションが必要以上にリソース競合解決を意識しないでも、システム全体として整合の取れた動作をおこなえるようにすることである。また、アプリケーションやリソースを制御するドライバの再利用性を高めることである。

【解決手段】 競合解決システムにおいて、リソースアクセス部 1 1 は、競合解決部 1 3 の結果を基に、アプリケーションがリソース部 1 6 にアクセスできるかどうかを判断し、アクセスできる場合はあらかじめリソース部 1 6 が登録してある関数を呼ぶことで、アプリケーションはリソース競合解決を意識しないで作成可能になり、また、リソース部 1 6 の再利用がしやすくなり、かつシステム全体として整合の取れた動作を行うことができる。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [000005821]

1. 変更年月日 1990年 8月28日

[変更理由] 新規登録

住 所 大阪府門真市大字門真1006番地
氏 名 松下電器産業株式会社